ISSN 1561-5359. Штучний інтелект, 2018, № 2

УДК 519.6

А.Н. Терещенко, В.К. Задирака
Институт кибернетики имени В.М. Глушкова НАН Украины, Украина
пр. Академика Глушкова, 40, г. Киев, 03680

ПАРАЛЛЕЛЬНОЕ СЛОЖЕНИЕ НА ОСНОВЕ ВЕКТОРНЫХ ОПЕРАЦИЙ

A.N. Tereshchenko, V.K. Zadiraka
Institute of Cybernetics of NAS of Ukraine, Ukraine
40, Academician Hlushkov av., Kyiv, 03680

PARALLEL ADDITION BASED ON VECTOR OPERATIONS

У роботі запропоновано новий метод реалізації операції багатослівного додавання у паралельній моделі обчислення. Запропонований метод базується на векторних операціях, що значно зменшує кількість задіяних процесорів. У роботі наведено алгоритм у паралельній моделі. Аналіз складності алгоритму показав, що у паралельній моделі обчислення кількість однослівних операцій $4k+59$ пропорційно залежить від кількості задіяних процесорів k за умови, що кожен процесор виконує однакову кількість векторних операцій довжини 16. Алгоритм реалізовано мовою програмування OpenCL (v.1.2) та протестовано.

Ключові слова: паралельна модель обчислення, векторні операції, багаторозрядна операція

The article describes the new method of implementation of multidigit addition in parallel model of computation. The method uses vector operations that reduces noticeably the number of stream processors. The article describes algorithm in parallel model of computation. The analysis of complexity of described algorithm shows that in parallel model of computation the number of one-digit operations $4k+59$ depends linearly on number of processors k used in calculation taking into account that every processor executes constant number of vector operations of the length of 16. Algorithm is implemented using language OpenCL (v. 1.2) and tested.

Keywords: parallel model of computation, vector operations, multidigit operation

Вступление

Микропроцессорная техника развивается очень быстро. Увеличение быстродействия за счет увеличения тактовой частоты процессора или числа процессоров ведет к значительному увеличению потребляемой мощности устройств, что ведет к дорогим последующим расходам на обслуживание таких устройств, управление процессорами, передачу данных между ними, написания программного обеспечения, которые сложно адаптируется при изменении числа задействованных процессоров. Объем электроэнергии, необходимый, например, на решение задачи учета ошибки округления при вычислении операций с трансвычислительной сложностью становится одним из факторов, который влияет как на выбор модели устройства (последовательная или параллельная), числа задействованных процессоров (в случае параллельной модели), так и алгоритма реализации вычислений в этой модели [1-5].

В данной работе рассматривается метод, который позволяет распараллелить многоразрядную операцию сложения. Предложенный метод базируется на векторных операциях, что значительно уменьшает число задействованных параллельных процессоров. Так, например, при параллельной обработке N элементов процессорами, которые поддерживают векторные операции длины 16, достаточно задействовать не более $\lceil N/16 \rceil$ процессоров.

Существует метод [1, 2], который позволяет сложить n-битные целые числа приблизительно за $O(k) = = 2\log_2 k$ шагов, если задействовать k параллельных вычислительных узлов $(k = n)$ и при условии, что одна 1-битная операция сложения занимает один такт (один шаг) процессора. Метод называется «методом с предсказыванием знака переноса» (carry-lookahead addition) и этот метод включен в

схемотехнику почти всех современных компьютеров.

При сложении двух mn-битных чисел на схеме потребуется $O(mn) = \log_2 m + \log_2 n$ шага. Далее показано, что данную операцию можно убыстрить, что актуально.

Постановка задачи

Даны mn-битные целые неотрицательные числа X и Y. Необходимо построить быстрый алгоритм вычисления суммы чисел X и Y в параллельной модели вычисления с использованием векторных операций.

Пусть X и Y представляют mn-битные целые неотрицательные числа, $n = km$. И пусть Z будет их $mn+1$-битная сумма. Число $X = \sum_{i=0}^{n-1}(X_i \cdot 2^{m \cdot i})$ представим в виде $\{X_{n-1},...,X_0\}$, где $X_i = \sum_{b=0}^{m-1}(x_{im+b} \cdot 2^b)$, $i = \overline{0, n-1}$ является m-битным словом. $X_i = \{x_{im+m-1},...,x_{im}\}$, $i = \overline{0, n-1}$. mn-битные числа Y и Z представляются таким же образом.

Далее, для удобства изложения, mn-битные числа X, Y, Z и другие будем называть n-словными числами, каждое слово которого является m-битным.

Найдем сначала сумму n-словных чисел X и Y без учета переноса знака между словами. Получим многословное число $M = \{M_{n-1},...,M_0\}$, где $M_i = \langle X_i + Y_i \rangle_{2^m}$, $i = \overline{0, n-1}$. На основании числа M длины n слов вычислим число T длины n бит, где в каждый бит записывается t_i, где

$$t_i = \begin{cases} 1 & if\ M_i = 2^m - 1 \\ 0 & if\ M_i \neq 2^m - 1 \end{cases}, \quad i = \overline{0, n-1}.$$

Для получения числа T сгруппируем биты t_i, $i = \overline{0, n-1}$, по m бит $T = \{T_{k-1},...,T_0\}$, где

$$T_p = \sum_{b=0}^{m-1}(t_i \cdot 2^b),$$

$$t_i = \begin{cases} 1 & if\ M_i = 2^m - 1 \\ 0 & if\ M_i < 2^m - 1 \end{cases}, \quad i = pm + b,$$

$$p = \overline{0, k-1}, \; k = \frac{n}{m}. \qquad (1)$$

Число T будет состоять из $k = \frac{n}{m}$ слов. Использование числа T позволяет спрогнозировать возникновение знака переноса в целой группе из m слов, на которые разбиваются n-словные чисел X и Y при многословном сложении.

Анализ генерирования знака переноса в группах из m слов

Рассмотрим следующее утверждение, которое позволяет определять знак переноса при сложении слов длины m бит без вычисления знака переноса, для нахождения которого обычным способом необходимо использование дополнительного слова для сохранения знака переноса.

Утверждение 1. Если числа X_i, Y_i, состоящие из m бит, такие, что $2^m \le X_i + Y_i$, то $(M_i < X_i) \vee (M_i < Y_i)$, где $M_i = \langle X_i + Y_i \rangle_{2^m}$.

Доказательство

Далее выражение, при котором $2^m \le X_i + Y_i$, будем называть условием, при котором возникает знак переноса.

Необходимо рассмотреть следующие случаи:

1. $X_i < \frac{2^m}{2}$, $Y_i < \frac{2^m}{2}$.

2. $X_i < \frac{2^m}{2}$, $\frac{2^m}{2} \le Y_i < 2^m$ или

   $Y_i < \frac{2^m}{2}$, $\frac{2^m}{2} \le X_i < 2^m$.

3. $\frac{2^m}{2} \le X_i < 2^m$, $\frac{2^m}{2} \le Y_i < 2^m$.

Случаи 1 и 3 являются простыми. В 1-м случае знак переноса никогда не

появляется $X_i + Y_i < \frac{2^m}{2} + \frac{2^m}{2} = 2^m$ и условие не выполняется:

$$(M_i < X_i) \vee (M_i < Y_i), \qquad (2)$$

где $M_i = \langle X_i + Y_i \rangle_{2^m}$,

Выполняется условие, при котором знак переноса не появляется:

$$(X_i \le M_i) \wedge (Y_i \le M_i), \qquad (3)$$

где $M_i = \langle X_i + Y_i \rangle_{2^m}$,

В 3-м случае числа X_i, Y_i можно представить следующим образом:

$$X_i = HX_i + \frac{2^m}{2},\ Y_i = HY_i + \frac{2^m}{2},$$

где $HX_i < \frac{2^m}{2}$, $HY_i < \frac{2^m}{2}$.

В этом случае при сложении чисел $X_i + Y_i$ всегда возникает знак переноса $HX_i + \frac{2^m}{2} + HY_i + \frac{2^m}{2} > 2^m - 1$. Сумма чисел по модулю $\langle X_i + Y_i \rangle_{2^m} = {=}\langle HX_i + HY_i + 2^m \rangle_{2^m} {=} \langle HX_i + HY_i \rangle_{2^m}$. Так как $HX_i < \frac{2^m}{2}$, $HY_i < \frac{2^m}{2}$ по определению, то после добавления HY_i или HX_i к правым и левым частям получаем, что

$$HX_i + HY_i < \frac{2^m}{2} + HY_i = Y_i,$$

$$HY_i + HX_i < \frac{2^m}{2} + HX_i = X_i.$$

Если $M_i = HX_i + HY_i$, то условие $(M_i < X_i) \wedge (M_i < Y_i)$ выполняется строже с условием «$\wedge$», вместо «$\vee$».

Рассмотрим 2-й случай, когда $X_i < \frac{2^m}{2}$, $\frac{2^m}{2} \le Y_i < 2^m$ или $Y_i < \frac{2^m}{2}$, $\frac{2^m}{2} \le X_i < 2^m$. Значения в числах X_i, Y_i можно поменять местами, поэтому для упрощения можно считать, что достаточно рассмотреть любой из двух вариантов. Выберем вариант $X_i < \frac{2^m}{2}$, $\frac{2^m}{2} \le Y_i < 2^m$. Представим число Y_i в виде $Y_i = HY_i + \frac{2^m}{2}$, где $HY_i < \frac{2^m}{2}$. Тогда $\langle X_i + Y_i \rangle_{2^b} = \left\langle X_i + HY_i + \frac{2^m}{2} \right\rangle_{2^b}$.

Рассмотрим два случая:

а) если $X_i + HY_i < \frac{2^m}{2}$, то знак переноса не появляется при $M_i = X_i + HY_i + \frac{2^m}{2}$. Условие (2) не выполняется. Выполняется условие (3).

б) если $\frac{2^m}{2} \le X_i + HY_i$, то знак переноса появляется при $M_i = X_i + HY_i - \frac{2^m}{2}$ и условие (2) выполняется, так как $M_i < Y_i$ или $X_i + HY_i - \frac{2^m}{2} < HY_i + \frac{2^m}{2}$, или $X_i < 2^m$.

Утверждение 1 доказано для всех случаев, что (2) выполняется, когда присутствует знак переноса, и не выполняется для остальных случаев.

Утверждение 2. Если числа X_i, Y_i, состоящие из m бит, такие, что $(M_i < X_i) \vee (M_i < Y_i), M_i = \langle X_i + Y_i \rangle_{2^m}$, то $2^m \le X_i + Y_i$.

Используя Утверждение 1, на основании n-словного числа M получим n-словное число C, состоящее из входящих знаков переноса, которые возникают при сложении слов $X_i + Y_i$,

$$c_{i+1} = \begin{cases} 1 & if\ (M_i < X_i) \vee (M_i < Y_i) \\ 0 & if\ (X_i \le M_i) \wedge (Y_i \le M_i) \end{cases},$$

$$i = \overline{0, n-2},\ c_0 = 0.$$

Знак

$$c_n = \begin{cases} 1 & if\ (M_{n-1} < X_{n-1}) \vee (M_{n-1} < Y_{n-1}) \\ 0 & if\ (X_{n-1} \le M_{n-1}) \wedge (Y_{n-1} \le M_{n-1}) \end{cases}$$

обозначим отдельно. Далее по тексту элементы, которые принимают или хранят знаки переноса предыдущих пословных слов, будем называть элементами, содержащими входящие знаки переносов, чтобы отличать от исходящих знаков переноса. Если из контекста понято, что речь идет об исходящем знаке переноса, то слово «исходящий» может опускаться.

Для получения k-словного числа C сгруппируем входящие знаки переноса c_i, $i = \overline{0, n-1}$, по m бит и запишем их в слова длиной m бит $C = \{C_{k-1}, ..., C_0\}$, где

$$C_p = \sum_{b=\begin{cases} 0 & if\ 0<p \\ 1 & if\ p=0 \end{cases}}^{m-1} (c_i \cdot 2^b),$$

$$c_i = \begin{cases} 1 & if\ 2^m \le X_{i-1} + Y_{i-1} \\ 0 & if\ X_{i-1} + Y_{i-1} < 2^m \end{cases},$$

$$i = pm + b,\ p = \overline{0, k-1},\ k = \frac{n}{m},$$

$$\langle C_0 \rangle_2 = 0. \qquad (4)$$

Числа $C = C_{k-1} ... C_0$ и $T = T_{k-1} ... T_0$ имеют одинаковую длину в n бит или $k = \dfrac{n}{m}$ слов.

Лемма 1. Если при сложении p-х групп из m-словных чисел $\{X_{pm+m-1}, ..., X_{pm}\}$ + $\{Y_{pm+m-1}, ..., Y_{pm}\}$ однословные числа C_p и T_p, полученные по формулам (1), (4) на основе чисел $\{X_{pm+m-1}, ..., X_{pm}\}$ и $\{Y_{pm+m-1}, ..., Y_{pm}\}$, такие, что $C_p + T_p \ge 2^m$, то в p-й группе появляется знак переноса.

Доказательство.

Не уменьшая общности, доказательство приведем для $m = 16$. Далее на рисунках x обозначает, что значение бита не важно. На рис. 1 представлен простой случай, когда генерируется знак переноса.

	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	C_p
	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	T_p
1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	

Рис. 1. Генерация знака переноса в p-й группе из m слов. Случай 1.

Старший бит m-битного числа C_p равный 1 говорит о том, что сумма слов, предшествующих старшим словам в p-й группе из m слов, $X_{pm+m-2} + Y_{pm+m-2} \ge 2^m$ (при $m = 16$, $X_{p16+14} + Y_{p16+14} \ge 2^m$). m-битное число C_p содержит биты входящих знаков переносов поэлементных сумм m-словных чисел $\{X_{pm+m-2}, ..., X_{pm-1}\} + \{Y_{pm+m-2}, ..., Y_{pm-1}\}$. Слова X_{pm-1} + Y_{pm-1} относятся к предыдущей группе p–1 из m слов.

Старший бит m-битного числа T_p равный 1 говорит о том, что сумма старших слов в p-й группе из m слов $X_{pm+m-1} + Y_{pm+m-1}$ равна $2^m - 1$ (при $m = 16$, $X_{p16+15} + Y_{p16+15} = 2^m - 1$). При добавлении любого ненулевого значения к $2^m - 1$ возникает знак переноса в следующей группе p+1 из m слов.

Далее на рис. 2 рассмотрим другой случай.

	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	C_p
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	T_p
1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	

Рис. 2. Генерация знака переноса в p-й группе из m слов. Случай 2

Младший бит числа C_p равный 1 говорит о том, что при сложении слов в предыдущей $p-1$ группе из m слов $X_{(p-1)m+r}+Y_{(p-1)m+r}$, $r=\overline{0,m-1}$, появляется знак переноса, который передается в p-ю группу из m слов.

Биты числа T_p равные 1 говорят о том, что все пословные суммы в p-й группе из m слов равны 2^m-1, $X_{pm+r}+Y_{pm+r}=2^m-1$, $r=\overline{0,m-1}$. Очевидно, что при добавлении любого ненулевого значения к сумме слов $X_{pm}+Y_{pm}=2^m-1$ приводит к тому, что возникает знак переноса для всей группы p из $m=16$ слов, что отражает рис. 2.

Таким образом, если у однословных чисел C_p и T_p, полученных на основе m-словных чисел $\{X_{pm+m-1},...,X_{pm}\}$ и $\{Y_{pm+m-1},...,Y_{pm}\}$ из p-й группы, согласно (1) и (4), какие-либо соответствующие биты равны 1 и все последующие старшие биты числа T_p также равняются 1, то это будет говорить о том, что в p-й группе при сложении генерируются знак переноса. В этом случае C_p+T_p будет больше или равно 2^m.

Для полноты доказательства далее на рис. 3 рассмотрим случай, когда соответствующие биты C_p и T_p не могут быть одновременно равными 1.

	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	C_p
	1	1	1	1	1	1	1	1	1	1	1	1	1	0	y	y	T_p
0	1	1	1	1	1	1	1	1	1	1	1	1	1	t	z	y	

Рис. 3. Знак переноса в p-ой группе из m слов не появляется.

Биты m-битного числа C_p равные 1 говорят о том, что при пословном сложении первых двух слов в группе p из m слов $X_{pm}+Y_{pm}$ и $X_{pm+1}+Y_{pm+1}$ генерируются исходящие знаки переноса пословно, которые заносятся в 1-й и 2-й биты m-битного числа C_p.

Биты m-битного числа T_p равные 1 говорят о том, что в группе p из m слов пословные суммы равны 2^m-1, $X_{pm+r}+Y_{pm+r}=2^m-1$, $r=\overline{3,m-1}$. 2-й бит числа T_p равный 0 говорит о том, что $X_{pm+2}+Y_{pm+2}<2^m-1$.

В случае, когда $y=1$, выражение C_p+T_p больше или равно 2^m, что невозможно, так как 2-й бит числа T_p нулевой (т.е. $X_{pm+2}+Y_{pm+2}<2^m-1$). Даже, если сумма слов $X_{pm+1}+Y_{pm+1}$ больше или равна 2^m и появляется знак переноса, который добавляется к последующей пословной сумме $X_{pm+2}+Y_{pm+2}+1<2^m$, то этого недостаточно, чтобы появился знак переноса во 2-й пословной сумме.

Покажем, что на рис. 3 значение y всегда будет равно 0, так как 1-й и 2-й биты C_p равны 1. Так как 1-й бит в C_p

равен 1, то сумма $X_{pm+0}+Y_{pm+0}$ больше или равна 2^m. Напомним, что каждый бит m-битного числа C_p содержит входящие знаки переноса. Для того, чтобы нулевой бит в T_p равнялся 1 необходимо, чтобы выполнялось условие, что $\left\langle X_{pm+0}+Y_{pm+0}\right\rangle_{2^m}=2^m-1$,что невозможно при начальном условии, что сумма слов $X_{pm+0}+Y_{pm+0}$ больше или равна 2^m (1-й бит в C_p равен 1 на рис. 3). Так сумма по модулю наибольших значений равна 2^m-2, $\left\langle 2^m-1+2^m-1\right\rangle_{2^m}=2^m-2$, что меньше 2^m-1.

С учетом того, что $y=0$, то $z=1$ на рис. 3. Значение t зависит от суммы чисел $X_{pm+1}+Y_{pm+1}$.

$$t=\begin{cases}1 & if\ X_{pm+1}+Y_{pm+1}+1=2^m-1\\ 0 & if\ X_{pm+1}+Y_{pm+1}+1<2^m-1\end{cases}.$$

Лемма доказана.

Лемма 1 является обобщением функции $g(\cdot)$ (*generate* – генерации знака переноса [1]) на векторные операции.

Лемма 2. Если при сложении p-х групп из m-словных чисел $\{X_{pm+m-1},...,X_{pm}\}+\{Y_{pm+m-1},...,Y_{pm}\}$ однословные числа C_p и T_p, полученные по формулам (1), (4) на основе чисел $\{X_{pm+m-1},...,X_{pm}\}$ и $\{Y_{pm+m-1},...,Y_{pm}\}$, такие, что $T_p=2^m-1$, то в p-й группе возникает знак переноса, если знак переноса возникает в предыдущей $p-1$ группе из m слов.

Доказательство.

Не уменьшая общности, доказательство приведем для $m=16$. Далее на рис. 4 представлен простой случай, когда знак переноса, полученный в $p-1$ группе из m слов передается "транзитно" в $p+1$ группу из m слов.

$$\begin{array}{ccccccccccccccccc} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \quad C_p\\ & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \quad T_p\\ \hline 1 & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x & x\end{array}$$

Рис. 4. Перенос знака из $p-1$ группы в $p+1$ группу из m слов.

Нулевые значения бит m-битного числа C_p говорят о том, что при пословном сложении $X_{pm+r-1}+Y_{pm+r-1}<2^m$, $r=\overline{0,m-1}$, т.е. знаки переноса не появляются.

Биты числа T_p равные 1 говорят о том, что суммы слов равны 2^m-1, $X_{pm+r}+Y_{pm+r}=2^m-1$, $r=\overline{0,m-1}$. Это случай, когда сумма чисел $\{X_{pm+m-1},...,X_{pm}\}$, $\{Y_{pm+m-1},...,Y_{pm}\}$ дает число, все биты которого равны 1. При добавлении знака переноса из $p-1$ группы из m слов получаем число, все биты которого равны 0, и при этом возникает знак переноса, который передается в $p+1$ группу из m слов.

Лемма доказана.

Лемма 2 является обобщением функции $p(\cdot)$ (*propagate* – размножение знака переноса [1]) на векторные операции.

Лемма 3. Если при сложении p-х групп из m-словных чисел $\{X_{pm+m-1},...,X_{pm}\}$ + $\{Y_{pm+m-1},...,Y_{pm}\}$ однословные числа C_p и T_p, полученные по формулам (1), (4) на основе чисел $\{X_{pm+m-1},...,X_{pm}\}$ и $\{Y_{pm+m-1},...,Y_{pm}\}$, такие, что $C_p+T_p<2^m$, то в p-й группе знак переноса не появляется.

Доказательство

Не уменьшая общности, доказательство рассмотрим для $m=16$. Далее на рис. 5 представлен случай, когда знак переноса, полученный в $p-1$ группе из m слов, передается в текущую группу p из m слов.

	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	C_p
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	T_p
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

Рис. 5. Знак переноса в p группе из m слов не появляется

Младший бит в m-битном числе C_p равный 1 говорит о том, что при сложении слов $X_{(p-1)m+r-1}+Y_{(p-1)m+r-1}$, $r=\overline{0,m-1}$, $p-1$ группы из m слов генерируется знак переноса, который передается в p группу из m слов.

Биты m-битного числа T_p равные 1 говорят о том, что в группе p из m слов пословные суммы $X_{pm+r}+Y_{pm+r}=2^m-1$, $r=\overline{1,m-1}$. 0-й бит числа T_p равный 0 говорит о том, что $X_{pm+0}+Y_{pm+0}\neq 2^m-1$. Эта сумма «поглощает» знак переноса, который появляется в группе $p-1$ из m слов.

На следующем рис. 7 показано, что в случае, когда соответствующие биты чисел C_p и T_p не равны 1, то знаки переноса «поглощаются».

На рис.7 C_p+T_p не превышает 2^m. Лемма доказана.

	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	C_p
	0	1	1	0	0	0	0	1	1	1	1	1	1	1	1	0	T_p
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

Рис. 7. Знаки переноса «поглощаются» в группе p из m слов.

Лемма 3 является обобщением функции $k(\cdot)$ (*kill* – «поглощение» знака переноса [1]) на векторные операции.

Примечание для Лемм 1, 2, 3. Так как в p-й группе из m слов сумма старших слов $X_{pm+m-1}+Y_{pm+m-1}$ может определять знак переноса в $p+1$ группу из m слов, то считаем, что сумма $X_{pm+m-1}+Y_{pm+m-1}\leq 2^m-1$ не генерирует знак переноса сама по себе.

На основании доказанных Лемм предлагается алгоритм «предсказания» знаков переноса пословного сложения n-словных чисел, с длиной слова в m бит, в последовательной модели вычислений.

Алгоритм 1. «Предсказание» знака переноса при сложении чисел длиной nm бит.

Вход: Числа $X=\{X_{n-1},...,X_0\}$, $Y=\{Y_{n-1},...,Y_0\}$, $k=\dfrac{n}{m}$.

Выход: $C=\{C_{k-1},...,C_0\}$, $T=\{T_{k-1},...,T_0\}$, $M=\{M_{n-1},...,M_0\}$, c–знак переноса.

Шаг 1. $c\leftarrow 0$.

Шаг 2. Если $X_{n-1} + Y_{n-1} \geq 2^m$, то $c \leftarrow 1$.

Шаг 3. Вычислить число $M = \{M_{n-1},...,M_0\}$, где $M_i = \langle X_i + Y_i \rangle_{2^m}$, $i = \overline{0, n-1}$.

Шаг 4. Вычислить число $T = \{T_{k-1},...,T_0\}$, где $T_p = \sum_{b=0}^{m-1}(t_i \cdot 2^b)$, $t_i = \begin{cases} 1 & if\ M_i = 2^m - 1 \\ 0 & if\ M_i < 2^m - 1 \end{cases}$, $i = pm + b$, $p = \overline{0, k-1}$, $k = \dfrac{n}{m}$.

Шаг 5. Вычислить число $C = \{C_{k-1},...,C_0\}$, где $C_p = \sum_{b=\begin{cases} 0 & if\ 0<p \\ 1 & if\ p=0 \end{cases}}^{m-1}(c_i \cdot 2^b)$,

$c_i = \begin{cases} 1 & if\ (M_{i-1} < X_{i-1}) \vee (M_{i-1} < Y_{i-1}) \\ 0 & if\ (X_{i-1} \leq M_{i-1}) \wedge (Y_{i-1} \leq M_{i-1}) \end{cases}$, $i = pm + b$, $p = \overline{0, k-1}$, $k = \dfrac{n}{m}$.

Шаг 6. Для $p \leftarrow 0$ до $p < k - 2$

Шаг 7. Если $C_p + T_p \geq 2^m$, то $C_{p+1} \leftarrow C_{p+1} \vee 1$.

Шаг 8. Конец цикла по p.

Шаг 9. Если $C_{k-1} + T_{k-1} \geq 2^m$, то $c \leftarrow 1$.

Теорема 1. Число однословных операций в Алгоритме 1 имеет вид $O(n) = 6n + 3k - 2$, где $n = km$ – число слов в каждом слагаемом, m – длина слова в битах.

Доказательство. Будем считать, что все однословные операции сложения и сравнения выполняются за одинаковое время. Тогда Шаг 2 занимает 1 операцию. Для вычисления числа M на Шаге 3 необходимо выполнить n операций. Для вычисления числа T на Шаге 4 необходимо выполнить n сравнений и столько же битовых операций. Если считать, что выполнение каждой битовой операции занимает столько же времени, как и операция однословного сравнения, то на выполнение Шага 4 необходимо $2n$ операции. Для вычисления числа C на Шаге 5 необходимо выполнить $2n$ операции однословного сравнения и n побитовых операций, что составляет $3n$ операции. На Шаге 7 сложение $C_p + T_p$ является двухсловной операцией, так как возможно переполнение результата сложения. Для того, чтобы уменьшить сложность вычислений, операцию вида $A + B \geq 2^m$ можно заменить следующими операциями: $M \leftarrow \langle A + B \rangle_{2^m}$, $(M < A) \vee (M < B)$, согласно доказанному ранее утверждению, что потребует 3 операции. Таким образом, для выполнения цикла на Шаге 7 понадобиться $3(k-2)$ операции. Шаг 9 также потребует 3 однословных операций, аналогично Шагу 7.

Общее число однословных операций, необходимых для выполнения Алгоритма 1, составляет $1 + n + 2n + 3n + 3(k-2) + 3 = 6n + 3k - 2$.

Теорема доказана.

Вычисление знака переноса n-битных чисел $C + T$ дает значение знака переноса суммы nm-битных чисел X и Y (простой случай, когда при сумме старших слов чисел X и Y появляется знак переноса, не берем во внимание, так как с этого случая начинается выполнение Алгоритма 1). По сути, задача вычисления знака переноса при сложении nm-битных чисел сведена к задаче вычисления знака переноса суммы n-битных чисел, что в m раз меньше начальной длины. В свою

очередь, вычисления знака переноса суммы n-битных чисел $C+T$ может быть сведено к задаче вычисления знака переноса суммы чисел длиной в $\frac{n}{m}$ бит и так далее за счет рекурсивного вызова Алгоритма 1. Таким образом, число уровней рекурсивного вызова Алгоритма 1 определяется выражением $\log_m n$, где m – длина слова в битах и n – длина числа в словах.

Так, например, если $n = km$, то число уровней вызова Алгоритма 1 будет равно $\log_m n = \log_m km = \; = 1 + \log_m k$. Если $k < m$, то $\lfloor 1 + \log_m k \rfloor = 1$, то рекурсивный вызов Алгоритма 1 не требуется.

Если $n = km^2$ и $k < m$, то $\lfloor \log_m n \rfloor = \lfloor \log_m km^2 \rfloor = \lfloor 2 + \log_m k \rfloor = 2$, что говорит о том, что нужен один дополнительный уровень рекурсивного вызова Алгоритм 1.

Вычисление знака переноса пословных сумм

Для вычисления общей суммы nm-битных чисел необходимо также найти знаки переноса пословных сумм.

Пословные операции над C_p и T_p позволяют вычислить знак переноса для следующей $p+1$ группы из m слов. Для этого вернемся к рис. 2. и рассмотрим его в следующем виде.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	C_p
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	T_p
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	C_p

Рис. 8. Корректировка пословных входящих знаков переноса C_p (в знаменателе) на основе входящих знаков переноса C_p (в числителе) в группе p из $m = 16$ слов

Младший бит в C_p равный 1 говорит о том, что при сумме $p-1$ группы из m слов $X_{(p-1)m+r} + Y_{(p-1)m+r}$, $r = \overline{0, m-1}$, появляется знак переноса, который передается в p группу из m слов и должен быть учтен при сложении $X_{pm+0} + Y_{pm+0} + 1$.

Биты m-битного числа T_p равные 1 говорят о том, что в группе p из m слов пословные суммы равны $X_{pm+r} + Y_{pm+r} = 2^m - 1$, $r = \overline{0, m-1}$. Очевидно, что при добавлении знака переноса к сумме $X_{pm+0} + Y_{pm+0} + 1$ приводит к тому, что возникает знак переноса во всех словах группы p из m слов, что показано на рис. 8. Напомним, что C_p, $p = \overline{0, n-1}$, содержит входящие знаки переносов.

Для нахождения знаков переносов для каждого слова необходимо использовать побитовые операции. Условие, при котором соответствующие биты двух m-битных слов C_p и T_p равны 1, говорит о том, что знак переноса добавляется к значению $2^m - 1$ и новый знак должен переноситься в следующее слово. Предлагается следующая побитовая операция для итерационной корректировки входящих знаков переноса $C_p \leftarrow C_p \vee \langle (C_p \wedge T_p) << 1 \rangle_{2^m}$, где побитовая операция «$<<1$» сдвигает значения битов влево (в сторону старших битов) на один бит. Далее на рис. 9 показан результат одной итерации с использованием побитовой операции.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	C_p
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	T_p
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	C_p

Рис. 9. Корректировка входящих знаков переноса C_p (в знаменателе) на основе входящих знаков переноса C_p (в числителе) в группе p из $m = 16$ слов

Для нахождения остальных бит числа C_p необходимо выполнить дополнительно 14 итераций.

Вычисление пословных сумм

Для нахождения пословных сумм с учетом пословных знаков переноса предлагается Алгоритм 2, который оперирует числами $C = \{C_{k-1},...,C_0\}$, $T = \{T_{k-1},...,T_0\}$, $M = \{M_{n-1},...,M_0\}$, c, полученными в результате работы Алгоритма 1.

Алгоритм 2. Вычисление пословных сумм с учетом пословных знаков переноса.

Вход: $C = \{C_{k-1},...,C_0\}$, $T = \{T_{k-1},...,T_0\}$, $M = \{M_{n-1},...,M_0\}$, c – знак переноса.

Выход: $Z = \{Z_{n-1},...,Z_0\}$.

Шаг 1. Для $p \leftarrow 0$ до $p < k-1$

Шаг 2. Для $r \leftarrow 0$ до $r < m-1$

Шаг3. $C_p \leftarrow C_p \vee \left\langle (C_p \wedge T_p) << 1 \right\rangle_{2^m}$.

Шаг 4. Конец цикла по r.

Шаг 5. Конец цикла по p.

Шаг 6. Для $p \leftarrow 0$ до $p < k-1$

Шаг 7. $c_{temp} \leftarrow C_p$.

Шаг 8. Для $r \leftarrow 0$ до $r < m-1$

Шаг 9. $Z_{pm+r} \leftarrow \left\langle M_{pm+r} + \left\langle c_{temp} \right\rangle_2 \right\rangle_{2^m}$

Шаг 10. $c_{temp} \leftarrow c_{temp} >> 1$.

Шаг 11. Конец цикла по r.

Шаг 12. Конец цикла по p.

Теорема 2. Число однословных операций в Алгоритме 2 имеет вид $O_2(n) = 6km + k$, где $n = km$ – число слов в каждом слагаемом, m– длина слова в битах.

Доказательство. Будем считать, что все однословные операции сложения и сравнения выполняются за одинаковое время. Шаг 3 занимает 3 побитовые операции в циклах, которые вычисляют знаки переноса для каждого слова. Для завершения циклов на Шагах 1, 2, 3, 4, 5 необходимо $3km$ побитовых операций. На Шаге 7 в цикле выполнится k операций. На Шагах 9, 10 необходимо выполнить 2 побитовые операции и одну операцию сложения в циклах, которые находят суммы слов с учетом знаков переноса для каждого слова. Для завершения циклов на Шагах 6, 8, 9, 10, 11, 12 необходимо выполнить $3km$ операции.

Общее число однословных операций, необходимых для выполнения Алгоритма 2, составляет $3km + k + 3km = 6km + k$.

Теорема доказана.

Циклы на Шагах 1-5 и Шагах 6-12 могут быть объединены, так как они содержат одинаковые параметры для p и r. Циклы разделены, чтобы показать, что Алгоритм 2 состоит из двух логических частей. Первая часть вычисляет знаки переносов для каждого слова и вторая часть находит окончательные значения слов результата с учетом рассчитанных пословных знаков переноса.

Сложение многословных чисел в параллельной модели вычислений.

Разобьем n-словные числа X и Y на группы по 16 слов, тогда числа X и Y можно представить следующим образом: $X = \{X16_{k-1},...,X16_0\}$, $Y = \{Y16_{k-1},...,Y16_0\}$, где $X16_p$, $Y16_p$, $p = \overline{0,k-1}$, $k = \frac{n}{16}$. Далее в алгоритме используются векторные операции вида $M16 \leftarrow \left\langle X16_p + Y16_p \right\rangle_{2^m}$, что равнозначно

выполнению цикла $m_i \leftarrow \langle x_i + y_i \rangle_{2^m}$, $i = \overline{0{,}15}$, где m_i, x_i, y_i – соответствующие элементы векторов $M16$, $X16_p$, $Y16_p$. Результатом операции $M16 \leftarrow \langle X16_p + Y16_p \rangle_{2^m}$ будет вектор элементов, который будет содержать поэлементную сумму по модулю 2^m ($m = 16$) векторов $X16_p$, $Y16_p$.

$C16[0]$ – нулевой элемент в векторе $C16$.

$C16 \leftarrow C16 << 1$ – сдвиг элементов в векторе на одну позицию в сторону старших элементов:

$$C16[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15] \leftarrow C16[0,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]$$

$C16 \leftarrow \begin{cases} G16 & if\ C16 = 1 \\ 0 & if\ C16 = 0 \end{cases}$ – эту операцию можно переписать в виде цикла:

$$C16[i] \leftarrow \begin{cases} G16[i] & if\ C16[i] = 1 \\ 0 & if\ C16[i] = 0 \end{cases},\ i = \overline{0{,}15}.$$

$C8 \leftarrow H(C16) + L(C16)$ – операции $H(C16)$ и $L(C16)$ обозначают вектора из 8 элементов, которые содержат 8 старших и 8 младших элементов вектора $C16$.

$C16[i]$, $C16_i$ – указывают на один и тот же элемент вектора $C16$.

Далее представлен алгоритм сложения, который выполняется в ядре параллельного процессора, при распараллеливании многословной операции сложения.

Алгоритм 3. Ядро вычисления операции сложения с использованием векторных операций длины 16 и операций над словами длиной $m = 16$ бит.

Вход: Числа $X = \{X16_{k-1}, ..., X16_0\}$,

$Y = \{Y16_{k-1}, ..., Y16_0\}$,

где $k = \dfrac{n}{16}$, p – номер процессора, $G16 = \{G_{15}, ..., G_0\}$, $G_i = 2^i$, $i = \overline{0{,}15}$.

Выход: $C = \{C_k, ..., C_0\}$,

$Z = \{Z16_{k-1}, ..., Z16_0\}$.

Шаг 1. $C_p \leftarrow 0$.

Шаг 2. $C_k \leftarrow 0$.

Шаг 3. $M16 \leftarrow \langle X16_p + Y16_p \rangle_{2^{16}}$.

Шаг 4.

$$C16 \leftarrow \begin{cases} 1 & if\ (M16 < X16_p) \vee (M16 < Y16_p) \\ 0 & if\ (X16_p \le M16) \wedge (Y16_p \le M16) \end{cases}$$

Шаг 5. $C_{p+1} \leftarrow C16[15]$.

Шаг 6.

$$C16[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15] \leftarrow C16[0,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]$$

Шаг 7. $C16[0] \leftarrow 0$.

Шаг 8. $C16 \leftarrow \begin{cases} G16 & if\ C16 = 1 \\ 0 & if\ C16 = 0 \end{cases}$.

Шаг 9. $T16 \leftarrow \begin{cases} G16 & if\ M16 = 2^{16} - 1 \\ 0 & if\ M16 < 2^{16} - 1 \end{cases}$.

Шаг 10.

$$C4 \leftarrow L(L(C16)) + H(L(C16)) + + L(H(C16)) + H(H(C16)).$$

Шаг 11.

$$T4 \leftarrow L(L(T16)) + H(L(T16)) + + L(H(T16)) + H(H(T16)).$$

Шаг 12.

$$C1 \leftarrow L(L(C4)) + H(L(C4)) + + L(H(C4)) + H(H(C4)).$$

Шаг 13.

$$T1 \leftarrow L(L(T4)) + H(L(T4)) + + L(H(T4)) + H(H(T4)).$$

Шаг 14. $F1 \leftarrow C1$.

Шаг 15. Для $i \leftarrow 0$ до $i < k - 1$

Шаг 16. $F1 \leftarrow F1 \vee \langle C_p \rangle_2$.

Шаг 17.

$C_{temp} \leftarrow \begin{cases} 1 & if\ 2^{16} \le F1 + T1 \\ 0 & if\ F1 + T1 < 2^{16} \end{cases}$ (или $C_{temp} \leftarrow \begin{cases} 1 & if\ (FT1 < F1) \vee (FT1 < T1) \\ 0 & if\ (F1 \le FT1) \wedge (T1 \le FT1) \end{cases}$, где $FT1 \leftarrow \langle F1 + T1 \rangle_{2^{16}}$).

Шаг 18. ДОЖДАТЬСЯ ОСТАЛЬНЫХ ПРОЦЕССОРОВ

Шаг 19. $C_{p+1} \leftarrow C_{p+1} \vee C_{temp}$.

Шаг 20. Конец цикла по i.

Шаг 21. ДОЖДАТЬСЯ ОСТАЛЬНЫХ ПРОЦЕССОРОВ

Шаг 22. $C1 \leftarrow C1 \vee C_p$.

Шаг 23. Для $i \leftarrow 0$ до $i < 16$

Шаг 24. $C1 \leftarrow C1 \vee \langle (C1 \wedge T1) << 1 \rangle_{2^{16}}$.

Шаг 25. Конец цикла по i.

Шаг 26. $C16 \leftarrow \begin{cases} 1 & if\ 0 < (C1 \wedge G16) \\ 0 & if\ (C1 \wedge G16) = 0 \end{cases}$.

Шаг 27. $Z16_p \leftarrow C16 + M16$.

Пояснения к Алгоритму 3

По завершению Алгоритма 3 $Z = \{Z16_{k-1},...,Z16_0\}$ будет содержать сумму чисел $X = \{X16_{k-1},...,X16_0\}$, $Y = \{Y16_{k-1},...,Y16_0\}$, а значение в элементе C_k вектора $C = \{C_k,...,C_0\}$ будет содержать знак переноса, если он возник при сложении чисел.

Значения $X = \{X16_{k-1},...,X16_0\}$, $Y = \{Y16_{k-1},...,Y16_0\}, Z = \{Z16_{k-1},...,Z16_0\}$, $C = \{C_k,...,C_0\}$, $G16 = \{G_{15},...,G_0\}$ находятся в глобальной памяти. Каждый из k процессоров имеет доступ ко всем элементам: некоторые только на чтение, некоторые на запись, а некоторые и на чтение, и на запись.

Массив элементов $C = \{C_k,...,C_0\}$ используется k процессорами для передачи знака переноса между собой. Элемент C_k по завершению Алгоритма 3 содержит знак переноса, который возникает при сложении чисел $X = \{X16_{k-1},...,X16_0\}, Y = \{Y16_{k-1},...,Y16_0\}$Элементы вектора $C = \{C_k,...,C_0\}$, C_p, $p = \overline{1, k-1}$ содержат входящие знаки переноса. В C_p записывается 1, если при сложении процессором $p-1$ группы из 16 слов возникает знак переноса. Процессор за номером $p = 0$ обрабатывает самую младшую нулевую группу из 16 слов, в которую не передается входящий знак переноса, поэтому $C_0 = 0$ всегда.

Каждый p-й процессор из k процессоров работает со своим входным вектором $X16_p$, $Y16_p$ и результат сложения также записывает в свой выходной вектор $Z16_p$, $p = \overline{0, k-1}$.

«Областью видимости» вектора $M16$, $C16$, $T16$ является исполняемый процессор. Другие процессоры оперируют со своими векторами.

На Шаге 1, 2 происходит инициализация массива, который содержит знак переноса. Каждый p-й процессор инициализирует свой элемент C_p массива $C = \{C_k,...,C_0\}$.

На Шаге 3 вычисляются суммы элементов по модулю. Каждый процессор работает со своей группой слов $X16_p$, $Y16_p$, $p = \overline{0, k-1}$.

На Шаге 4 вычисляются исходящие знаки переноса. Каждый элемент вектора $C16$ содержит знак переноса для каждого слова. Значения знаков переносов не учитывают влияние сумм элементов из предшествующих по номеру векторов. На этом этапе элементы вектора $C16$, $C16[i]$, $i = \overline{0,15}$ содержат исходящие знаки переноса пословных (поэлементных) сумм векторов $X16_p$, $Y16_p$.

На Шаге 5 исходящий знак переноса, полученный при сложении самых старших элементов векторов $X16_p$, $Y16_p$, передается следующему процессору $p+1$ посредством массива $C = \{C_k,...,C_0\}$, который содержит входящие знаки переносов.

На Шаге 6 происходит сдвиг элементов, которые содержат знаки переносов, на одну позицию в сторону старших элементов. После сдвига исходящие знаки переноса становятся входящими и теперь можно воспользоваться Леммами 1, 2, 3 для предсказания знаков переносов в каждой группе из 16 слов, которые представлены своими векторами $M16$ для каждого процессора.

На Шаге 7 обнуляется элемент с

индексом 0 в векторе $C16$.

На Шаге 8 элементы $G16 = G_{15}...G_0$ используются только в режиме чтения и необходимы для того, чтобы преобразовать вектор длины 16 ($C16$, $T16$) в слово ($C1, T1$) из 16 бит, каждый бит которого равен 1, если соответствующий элемент вектора имеет ненулевое значение. Так, например, элементы вектора $C16 = \{C_{15},...,C_0\} =$ $= \{0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0\}$ после Шага 7 преобразуются в следующие значения

$$C16 = \{0,0,0,0,0,0,0,0,0,0,0,16,8,0,2,0\}$$

$G16 = \{G_{15},...,G_0\}$ также используется для обратного преобразования, когда записываются единицы в элементы векторов, если соответствующие биты слова не равны нулю.

На Шаге 8 знаки переносов преобразовываются в биты согласно индексам элементов вектора $C16$.

На Шаге 9 аналогичное преобразование происходит для вектора $T16$.

На Шаге 10 вектор $C16$ из 16 элементов разбивается на 4 группы по 4 слова. Все группы складываются поэлементно. Так, например, если вектор $C16$ содержит элементы $C16 = \{0,0,0,0,0,0,0,0,0,0,0,16,8,0,2,0\}$,то вектор $C4$ будет содержать элементы со значениями $C4 = \{8,0,2,16\}$.

На Шаге 11 аналогично $C4$ вычисляются значения для вектора $T4$.

«Упаковка пословных входящих знаков переноса» в битах одного слова на Шагах 10, 11, 12, 13 и «распаковка пословных входящих знаков переноса» из битов одного слова на Шаге 26 являются основной идеей Алгоритма 3. После «упаковки пословных входящих знаков переноса» можно оперировать всеми пословными знаками переноса одной группы из 16 слов одновременно в одной однословной операции. Однословная операция сложения не применима в этом случае. Так, например, если входящие знаки переносов при пословном сложении равны $C16 = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1\}$, а $T16 = \{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1\}$, то откорректированные входящие знаки переносов будут следующими

$$\{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1\}.$$

На Шаге 12 происходит окончательная «упаковка пословных входящих знаков переноса» в одном слове $C1$ длиной в 16 бит.

На Шаге 13 аналогично $C1$ вычисляется $T1$, который необходим для «транзитного» перемещения знака переноса из одного слова в следующее.

На Шаге 14 значение $C1$ копируется в $F1$, который используется для вычисления исходящего знака переноса группы p из m слов с учетом входящего знака переноса из предыдущей группы $p-1$.

На Шагах 15, 16, 17, 18, 19, 20 вычисляется знак переноса, который появляется в группе p из m слов. Вычисление происходит на основе слов $F1$, $T1$ и знака переноса переданного из предыдущей группы $p-1$. Число итераций является максимальным и равным k, так как для процессора с наибольшим индексом $k-1$ необходимо k шагов, пока знак переноса из самой младшей группы из m слов будет передан последовательно в самую старшую группу.

На Шагах 15, 16, 17, 18, 19, 20 выполняется вычисление исходящих знаков переносов с учетом «предсказания» входящих знаков переносов для каждой группы из m слов.

На Шаге 16 знаки переноса, которые появились в группе $p-1$ и переданы в группу p через элемент C_p, записываются в самый младший бит $F1$, который содержит копию входящих пословных знаков переноса.

На Шаге 17 на каждой итерации вычисляется бит исходящего пословного знака переноса (см. рис.9).

На Шаге 18 происходит синхронизация работы всех k параллельно

работающих процессоров по отношению к чтению значений элементов массива $C=\{C_k,...,C_0\}$, который используется для передачи знака переноса процессору с номером индекса большим на единицу.

На Шаге 19 знак переноса, возникший при сложении в группе p из m слов, которую обрабатывает $p-1$, процессор, передается следующему за индексом процессору через вектор $C=C_k...C_1$.

На Шаге 21 происходит синхронизация работы всех k параллельно работающих процессоров по отношению к записи в элементы массива $C=\{C_k,...,C_0\}$.

На Шаге 22 в слово $C1$, которое содержит входящие знаки переносов пословных слов, в самый младший бит добавляется знак переноса, который возник в предыдущей группе $p-1$. C_p и содержит значение «предсказанного» исходящего знака переноса группы $p-1$. Для p-й группы этот знак переноса будет являться входящим знаком переноса.

На Шагах 23, 24, 25 вычисляются пословные входящие знаки переносов внутри каждой группы p из $m=16$ слов с учетом вычисленного входящего знака переноса для группы p.

На Шаге 26 происходит «распаковка входящих знаков переноса», которые необходимы для окончательного вычисления пословных сумм внутри каждой группы p из $m=16$ слов.

На Шаге 27 пословная сумма корректируется вычисленными пословными входящими знаками переноса для получения окончательного результата. Исходящие знаки переноса вычислять и сохранять уже нет необходимости.

Теорема 3. Число однословных и векторных операций в Алгоритме 3 может быть представлено следующими выражениями

$$O_{однословных}(k)=4k+59,\ O_{векторных}(k)=15,$$

где k – число задействованных параллельных процессоров.

Доказательство. Считаем, что все однословные операции сложения, сравнения, побитовые однословные операции выполняются за одинаковое число тактов. Также считаем, что векторные операции сложения, сравнения, векторные побитовые операции также выполняются за одинаковое между собой время.

На Шагах 1,2 выполняется 2 однословные операции. Шаг 3 занимает одну векторную операцию, где длина вектора равна 16. На Шаге 3 операция вычисления значения по модулю $\left\langle X16_p+Y16_p\right\rangle_{2^{16}}$ выполняется автоматически при оперировании элементами вектора с типом данных в 16 бит. Для выполнения Шага 4 необходимо 2 векторные операции сравнения и одна векторная логическая операция. Шаги 5, 7 займут по одной однословной операции. На Шаге 6 выполняется одна векторная операция, результатом которой является пересортированный вектор. На Шагах 8, 9 выполняется по одной векторной операции сравнения. На каждом Шаге 10, 11 выполняются по 3 векторные операции сложения длины 4. На каждом из Шагов 12, 13 выполняется по 3 однословные операции сложения. На каждой итерации цикла на Шагах 15-20 выполняется по 2 логические однословные операции, одна однословная операция сравнения и одна однословная операция по модулю 2. Всего в цикле на Шагах 15-20 выполнится $4k$ однословных операций. На Шаге 22 выполняется одна однословная побитовая операция. На Шаге 24 выполняется 2 однословные побитовые операции и одна однословная побитовая операция сдвига. Всего в цикле на Шагах 23-25 выполняется $3\cdot16=48$ однословные побитовые операции. На Шагах 26, 27 выполняется одна векторная операция сравнения, одна векторная операция сложения с длиной вектора 16.

Общее число однословных операций выражается 2+2+3+3+$4k$+1+48= $=4k$+59. Общее число векторных операций выражается 1+3+1+2+3+3+2=15.

Теорема доказана.

Анализируя Теорему 3, видим, что число однословных операций не зависит от длины слагаемых, а зависит от числа процессоров, задействованных в многословной операции сложения в параллельной модели вычислений. Число векторных операций является константой и не зависит от длины слагаемых и числа задействованных процессоров.

Выводы

В данной работе предложен новый метод реализации операции многословного сложения в параллельной модели вычислений. Предложенный метод базируется на векторных операциях, что значительно уменьшает число задействованных процессоров и значительно уменьшает объем потребляемой мощности. В работе приведен алгоритм, на основании которого вычислительное ядро GPU реализует операцию сложения в параллельной модели вычислений на основе векторных операций. Анализ сложности приведенного алгоритма показал, что в параллельной модели вычислений число однословных операций $4k+59$ имеет линейную зависимость от числа задействованных процессоров k при условии, что процессоры выполняют константное число векторных операций длины 16. Результаты тестирования, проведенные при реализации алгоритма на языке программирования OpenCL версии 1.2, подтвердили корректность теоретических результатов.

Література

1. Catherine C. McGeoch, *Parallel Addition*, The American Mathematical Monthly, Vol. 100, No. 9 (Nov., 1993), pp. 867-871. Получено с (http://www.jstor.org/stable/2324666).
2. T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press 1990.
3. Терещенко А.Н., Задирака В.К. *Оценка сложности операции умножения многоразрядных чисел в параллельной модели вычислений* // Компьютерная математика. – 2016. – № 1.
4. Терещенко А.Н. *Аналіз складності операції множення багаторозрядних чисел при реалізації у паралельній моделі обчислень* // Проблеми програмування. – 2015. – № 1.
5. Терещенко А.М. *Оптимізація багаторозрядного множення на основі ШПФ у паралельній моделі обчислень* // Захист інформації. – 2014. – № 3. – С.178–184.

References

1. Catherine C. McGeoch, Parallel Addition, The American Mathematical Monthly, Vol. 100, No. 9 (Nov., 1993), pp. 867-871. (http://www.jstor.org/stable/2324666).
2. T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press 1990.
3. Tereshchenko A.N., Zadiraka V.K. *Otsenka slozhnosti operatsii umnozheniya mnogorazryadnyih chisel v parallelnoy modeli vyichisleniy* // Kompyuternaya matematika. – 2016. – # 1.
4. Tereshchenko A.N. *Analiz skladnosti operatsii mnozhennia bahatorozriadnykh chysel pry realizatsii u paralelnii modeli obchyslen* // Problemy prohramuvannia. – 2015. – № 1.
5. Tereshchenko A.M. *Optymizatsiia bahatorozriadnoho mnozennia na osnovi ShPF u paralelnii modeli obchyslen* // Zakhyst informatsii. – 2014. – № 3. – S.178–184.

RESUME

A.N. Tereshchenko, V.K. Zadiraka

Parallel addition based on vector operations

The article describes the new method of implementation of multidigit addition in parallel model of computation. The carry flag has been handled automatically in a sequential model of computation, so it does not have influence on overall complexity of the operation, but the carry flag is one of the main factors that influence the choice of algorithm and the number of processors involved in the parallel model of computation. The analysis of "generation", "promotion" and "killing" of carry flags for a group of words in the multidigit addition is carried out in this article. The carry flags could be combined into a group of bits the number equal to the length of one word in bits, and written in the multibit word *C* (criterion). It is proposed to use an additionnal criterion (word) *T*, which is calculated basing on the same group of words, on which the criterion (word) *C* is calculated. The lemmas, using an example of a group of 16 words, prove that the two criteria (words) *C* and *T* are sufficient to "predict" carry flag in a group of 16 words. The article describes and formulates the iterative correction of the input carry flags in the word *C* via bitwise opera-

tions using carry flags which are “predicted” in addition of the previous groups of word. The algorithm implements a "prediction" of the carry flags based on one-word bit operations and vector operations for groups of 16 words. The algorithm is described on the way to be implemented as a core of a parallel processor in parallel model of computation. The alogarithm logically consists of several parts: the transfer of the carry flag to the next group after adding highest words of the group; the calculation of the criteria *C* and *T*, which includes the "packing" of carry flags in the word *C*; iterative correction of carry flags which are “predicted” in addition of the previous groups of word; "unpacking" the corrected carry flags and adding them to values of the words of the group to get the final result. The use of "packing" / "unpacking" the carry flags of words of group and the use of vector operations greatly reduces the number of processors involved. The analysis of complexity of described algorithm shows that in parallel model of computation the number of one-digit operations depends linearly on number of processors used in calculation considering that processors support vector operations and every processor executes constant number of vector operations of the length of 16. Algorithm is implemented using language OpenCL (v. 1.2) and tested. The results of the test prove of theoretical data.

Надійшла до редакції 25.09.2018

